UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/687,941	10/17/2003	Abdul Kayam	24043-08537	9901

758 7590 05/28/2009

FENWICK & WEST LLP
SILICON VALLEY CENTER
801 CALIFORNIA STREET
MOUNTAIN VIEW, CA 94041

EXAMINER	
WANG, BEN C	
ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
05/28/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/687,941	KAYAM ET AL.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *24 April 2009*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-14, 17, 20-50, and 62-71* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☐ Claim(s) _____ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on April 24, 2009 has been entered.

2. Applicant's amendment dated April 24, 2009, responding to the Final Office action mailed November 24, 2008 provided in the rejection of claims 1-14, 17, 20-50, and 62-67, wherein claims 1, 36, and 63 have been amended and claims 68-71 have been newly added.

Claims 1-14, 17, 20-50, and 62-71 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims rejection have been fully considered but are moot in view of the new grounds of rejection – see *Beringer et al*, art made of record, as applied hereto.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-2, 17, 27, 36-41, 49-50, 63-65, and 67-71 are rejected under 35 U.S.C. 103(a) as being unpatentable over Beringer et al. (Pub. No. US 2004/0205765 A1) (hereinafter 'Beringer' - art made of record) in view of Kasi et al. (Pat. No. US 7,340,508 B1) (hereinafter 'Kasi')

4. **As to claim 1** (Currently Amended), Beringer discloses a method of creating an application for executing on at least one machine having a memory, the method comprising:

- creating a definition of at least one node (e.g., Fig. 3A, elements 37 - Transport Attribute; 38 - Process Attribute; [0032] - ... the transport type definition element specifies the possible values for transport attributes 37 ...; [0033]) and a specification (e.g., Fig. 3A, element 34 – Schema for Binding Element Extension; [0018] - ... a schema is provided which may be used by the service provider to define one or more web services ...; Fig. 1, elements 34 – SCHEMA; 18 – Web-Services Interface; [0020] - ... Service provider 12 publishes at least one web-services interface 18 based at least in part on a schema 34 (Fig. 3A). Web-services interface 18 preferably comprises a WSDL interface, for example a WSDL document ...), which are both held in at least one machine readable data file and written in a markup language (e.g., [0003] – WSDL (Web Services Description Language) is a web-services description language that describes

web-services by specifying parts, message, operations, ports, port types and services. It comprises an XML (eXtensible Markup Language) vocabulary that standardizes how organizations describe web-services …);

the specification being arranged to be processed by a run time environment (e.g., [0017] - … for defining a binding for web-services messages. The binding defines or specifies the features and/or format desirable in messages between the service provider and the service requester; [0108] – Figs. 5A-5D illustrate a flowchart of an exemplary method for processing … a message received by service provider …) and the specification defining:

i: how the at least one node interacts with other nodes during the processing of the specification (e.g., [0004] - … The BizTalk Messaging Framework specifies the format of a web-services message. It defines various SOAP header element, such as a 'process' element and a 'properties' element ...; [0005]; [0018] - … The messages may be web-services messages that are in accordance with an asynchronous messaging framework, such as BizTalk Messaging Framework);

ii: resources useable by the at least one node during the processing of the specification (e.g., [0026] - … the identifier is a Uniform Resource Locator …; [0034] - … Process attribute 38 is of type *anyURI*, which indicates that any Uniform Resource Identifier (URI) for a process element extension may be specified …);

iii: at least one set of predetermined rules used by the at least one node during the processing of the specification (e.g., [0041] - … a topic rule type definition element is used to define an element of type *topicRuleType* …; [0042]); and

iv: a set of messages which are arranged to be passed between nodes (e.g., [0017] - ... The WSDL document may also provide information on the types of message that may be exchanged between the service provider and a service requester ...) during the processing of the specification;

- causing the run time environment to process the specification (e.g., [0108] – Figs. 5A-5D illustrate a flowchart of an exemplary method for processing ...) held in the machine readable data file such that the at least one node, as defined therein, is implemented within the memory of the machine thereby becoming a memory resident node;

the at least one memory resident node being arranged to:

i. receive messages as defined within the specification (e.g., [0019] - ... When a service provider receives a message, such as a web-services document or an XML (eXtensible Markup Language) document, it can determine whether the message complies with the web services interface as defined by the service provider ...);

ii. process, according to rules defined in the specification for that node (e.g., [0019] - ... it can determine whether the message complies with the web services interface as defined by the service provider ...), data provided to the at least one memory resident node by messages such that rules are triggered if predetermined data is present within a message (e.g., Fig. 1, element – INVOKE; [0021] - ... invoke operations on web-services server ...; [0052] - ... specifies elements or data types that may be used for the application specific content ...; [0076] - ... A value of

'encoded' for the use attribute indicates that rules are used for encoding and the sender of the commitment receipt should obey the encoded rules; [0077]); and

iii. output further messages dependent upon triggering of rules (e.g., [0080] - ... An attachment extension in accordance with schema 46 may be specified in predefined locations in the WSDL document, for example 'binding/operation/output' ...) within the node; and

- the processing the specification, in the run time environment (e.g., [0108] – Figs. 5A-5D illustrate a flowchart of an exemplary method for processing ...)

Further, Beringer discloses a web-services interface for a web-service comprises a message binding extension element operable to specify a format for a message requesting the web service and a binding details extension element operable to specify an availability status of at least one receipt for the message (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Exposing Process Flows and Choreography Controllers as Web Services*, Kasi discloses:

- interconnecting the at least one memory resident node according to the specification and/or data input such that data input to the application created by processing of the specification is processed by the at least one memory resident node and, if further processing is required, forwarded to other nodes via links (e.g., Col. 4, Line 63 through Col. 5, Line 17 – Connectors may be labeled simple connectors, hubs or central connectors ... So-called hubs are used by connectors that are explicitly directed or linked to them ...; Col. 10, Lines 4-22 -

... a dispatcher in the connector can link the message to an existing process instance ...) for that processing;

- wherein links between nodes are dynamically configured responsive to amendments to the specification during processing thereof by the run time environment (e.g., Col. 7, Lines 37-43 – ... to dynamically determine some or all of document versions, service version, and choreography version ... a web service can be substantially updated and retain interoperability with other services that have not been similarly updated)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Kasi into the Beringer's system to further provide other limitations stated above in the Beringer system.

The motivation is that it would further enhance the Beringer's system by taking, advancing and/or incorporating the Kasi's system which offers significant advantages for supporting document exchange choreographies; and facilitating evolution of systems by various combinations of choreography versioning, service versioning and document versioning as once suggested by Kasi (e.g., Abstract)

5. **As to claim 2** (Previously Presented) (incorporating the rejection in claim 1), Kasi discloses the method in which a plurality of nodes are created (e.g., Col. 1, Lines 57-60 - ... as multiple context setting franchised services)

6. **As to claim 17** (Previously Presented) (incorporating the rejection in claim 1), Kasi discloses the method which comprises writing the messages in a flat text format,

which may be any of the following: ASCII, XML (e.g., Col. 2, Lines 33 - ... XML standards like XSDL, Xlink ...), EDI (Electronic Data Interchange) (e.g., Col. 1, Lines 62-64 - ... electronic data interchange (EDI))

7. **As to claim 27** (Previously Presented) (incorporating the rejection in claim 1), refer to claim **17** above accordingly.

8. **As to claim 36** (Currently Amended), Beringer discloses a computer system having a memory and being arranged to create an application, said system comprising:

- a node creator arranged to create at least one machine readable data file containing a definition (e.g., Fig. 3A, elements 37 - Transport Attribute; 38 - Process Attribute; [0032] - ... the transport type definition element specifies the possible values for transport attributes 37 ...; [0033]) of at least one node and a specification (e.g., Fig. 3A, element 34 – Schema for Binding Element Extension; [0018] - ... a schema is provided which may be used by the service provider to define one or more web services ...; Fig. 1, elements 34 – SCHEMA; 18 – Web-Services Interface; [0020] - ... Service provider 12 publishes at least one web-services interface 18 based at least in part on a schema 34 (Fig. 3A). Web-services interface 18 preferably comprises a WSDL interface, for example a WSDL document ...), the definition and specification each written in a markup language (e.g., [0003] – WSDL (Web Services Description Language) is a web-services description language that describes web-services by specifying parts,

message, operations, ports, port types and services. It comprises an XML (eXtensible Markup Language) vocabulary that standardizes how organizations describe web-services …);

the specification being arranged to be processed by a run time environment (e.g., [0017] - … for defining a binding for web-services messages. The binding defines or specifies the features and/or format desirable in messages between the service provider and the service requester; [0108] – Figs. 5A-5D illustrate a flowchart of an exemplary method for processing … a message received by service provider …) and the specification defining:

i: how the at least one node interacts with other nodes during the processing of the specification (e.g., [0004] - … The BizTalk Messaging Framework specifies the format of a web-services message. It defines various SOAP header element, such as a 'process' element and a 'properties' element ...; [0005]; [0018] - … The messages may be web-services messages that are in accordance with an asynchronous messaging framework, such as BizTalk Messaging Framework);

ii: resources useable by the at least one node during the processing of the specification (e.g., [0026] - … the identifier is a Uniform Resource Locator …; [0034] - … Process attribute 38 is of type *anyURI*, which indicates that any Uniform Resource Identifier (URI) for a process element extension may be specified …);

iii: at least one set of predetermined rules used by the at least one node during the processing of the specification (e.g., [0041] - … a topic rule type definition element is used to define an element of type *topicRuleType* …; [0042]); and

iv: a set of messages which are arranged to be passed between that node and any other node during the processing of the specification (e.g., [0019] - … When a service provider receives a message, such as a web-services document or an XML (eXtensible Markup Language) document, it can determine whether the message complies with the web services interface as defined by the service provider …)

Further, Beringer discloses a web-services interface for a web-service comprises a message binding extension element operable to specify a format for a message requesting the web service and a binding details extension element operable to specify an availability status of at least one receipt for the message (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Exposing Process Flows and Choreography Controllers as Web Services*, Kasi discloses:

- a linker arranged to connect, dynamically according to the specification and/or any data input to the application, at least two nodes such that data is arranged to pass between the nodes and the linker is arranged to interact with the node creator to modify the definition provided by the specification (e.g., Col. 4, Line 63 through Col. 5, Line 17 – Connectors may be labeled simple connectors, hubs or central connectors … So-called hubs are used by connectors that are explicitly directed or linked to them …; Col. 10, Lines 4-22 - … a dispatcher in the connector can link the message to an existing process instance …; Col. 7, Lines 37-43 – … to dynamically determine some or all of document versions, service

version, and choreography version ... a web service can be substantially updated and retain interoperability with other services that have not been similarly updated);

- a deployer arranged to deploy the application from the definition created by the node creator and the linker according to the specification wherein the run time environment (e.g., Col. 6, Lines 52-55 - At runtime, an intersection of supported choreographies among participating service instance is computed and a particular version of choreography is selected for use in the message exchange) is arranged to implement the nodes in the memory of the computer system, the at least one node thereby becoming a memory resident node and the at least one memory resident node being arranged to process data according to the rules (e.g., Col. 10, Lines 34-53 - ... The service visibility rules may include ...; Col. 11, Lines 17-21 – The composition agent correlates related messages by executing rules ...) wherein at least one of the rules is triggered if predetermined data is present and an output is generated from that memory resident node (e.g., Col. 9, Lines 27-29 – This context goes through a series of state transitions triggered by other events and the process itself can initiate events as part of the flow; Col. 5, Lines 18-31 - ... defines the input/output messages of the operations ... synchronous, with one inward message followed by one outward message ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Kasi into the Beringer's system to further provide other limitations stated above in the Beringer system.

The motivation is that it would further enhance the Beringer's system by taking, advancing and/or incorporating the Kasi's system which offers significant advantages for supporting document exchange choreographies; and facilitating evolution of systems by various combinations of choreography versioning, service versioning and document versioning as once suggested by Kasi (e.g., Abstract)

9. **As to claim 37** (Previously Presented) (incorporating the rejection in claim 36), Kasi discloses the computer system which comprises at least one processor arranged to process data, including the definition (e.g., Col. 6, Lines 1-20 – The present invention introduces the concept of a service definition version, which also can be dynamically determined ... A service definition has a name and version and defines the list of activities in the service along with one or more activity interface ...), and on which the definition created by the node creator and modified by the linker is processed (e.g., Col. 4, Line 63 through Col. 5, Line 17 – Connectors may be labeled simple connectors, hubs or central connectors ... So-called hubs are used by connectors that are explicitly directed or linked to them ...; Col. 10, Lines 4-22 - ... a dispatcher in the connector can link the message to an existing process instance ...)

10. **As to claim 38** (Previously Presented) (incorporating the rejection in claim 37), Kasi discloses the computer system which comprises at least one processing apparatus comprising the at least one processor and in which the linker is arranged to connect nodes running on the processor within the processing apparatus (e.g., Col. 4, Line 63

through Col. 5, Line 17 – Connectors may be labeled simple connectors, hubs or central connectors … So-called hubs are used by connectors that are explicitly directed or linked to them …; Col. 10, Lines 4-22 - … a dispatcher in the connector can link the message to an existing process instance …)

11. **As to claim 39** (Previously Presented) (incorporating the rejection in claim 38), Kasi discloses the computer system which comprises a plurality of processors, each remote from the other and having connector therebetween capable of transmitting data between the processors (e.g., Col. 4, Lines 28-46 – Connector is a general term for applications that communicate with other applications … Connectors that communicate on a directed basis communicate through hub connectors according to routing rules …)

12. **As to claim 40** (Previously Presented) (incorporating the rejection in claim 39), Kasi discloses the computer system in which each of the processors is provided on a separate processing apparatus (e.g., Col. 17, Lines 27-31 – If certain links in a service are hosted in a different application server or web server and need a separate web agent to protect them, these can either be grouped into a separate service, or registered as separate activities in the same service)

13. **As to claim 41** (Previously Presented) (incorporating the rejection in claim 39), Kasi discloses the computer system in which the linker is arranged to connect nodes provided on processors remote from one another (e.g., Col. 4, Line 63 through Col. 5,

Line 17 – Connectors may be labeled simple connectors, hubs or central connectors ... So-called hubs are used by connectors that are explicitly directed or linked to them ...; Col. 10, Lines 4-22 - ... a dispatcher in the connector can link the message to an existing process instance ...)

14. **As to claim 49** (Original) (incorporating the rejection in claim 1), refer to claim **1** above accordingly.

15. **As to claim 50** (Original) (incorporating the rejection in claim 36), refer to claim **36** above accordingly.

16. **As to claim 63** (Currently Amended), Beringer discloses a computer system having a memory and being arranged to run an application, said system comprising:

- a run time environment (e.g., [0108] – Figs. 5A-5D illustrate a flowchart of an exemplary method for processing ...) arranged to process a definition (e.g., Fig. 3A, elements 37 - Transport Attribute; 38 - Process Attribute; [0032] - ... the transport type definition element specifies the possible values for transport attributes 37 ...; [0033]) of at least one node and a specification (e.g., Fig. 3A, element 34 – Schema for Binding Element Extension; [0018] - ... a schema is provided which may be used by the service provider to define one or more web services ...; Fig. 1, elements 34 – SCHEMA; 18 – Web-Services Interface; [0020] - ... Service provider 12 publishes at least one web-

services interface 18 based at least in part on a schema 34 (Fig. 3A). Web-services interface 18 preferably comprises a WSDL interface, for example a WSDL document ...) which are both written in a markup language and held within a machine readable data file (e.g., [0003] – WSDL (Web Services Description Language) is a web-services description language that describes web-services by specifying parts, message, operations, ports, port types and services. It comprises an XML (eXtensible Markup Language) vocabulary that standardizes how organizations describe web-services ...), the specification defining:

i: how the at least one node interacts with other nodes during the processing of the specification (e.g., [0004] - ... The BizTalk Messaging Framework specifies the format of a web-services message. It defines various SOAP header element, such as a 'process' element and a 'properties' element ...; [0005]; [0018] - ... The messages may be web-services messages that are in accordance with an asynchronous messaging framework, such as BizTalk Messaging Framework);

ii: resources useable by the at least one node during the processing of the specification (e.g., [0026] - ... the identifier is a Uniform Resource Locator ...; [0034] - ... Process attribute 38 is of type *anyURI*, which indicates that any Uniform Resource Identifier (URI) for a process element extension may be specified ...);

iii: at least one set of predetermined rules (e.g., [0041] - ... a topic rule type definition element is used to define an element of type *topicRuleType* ...; [0042]) used by the at least one node during the processing of the specification; and

iv: a set of messages which are arranged to be passed between nodes during the processing of the specification (e.g., [0017] - ... The WSDL document may also provide information on the types of message that may be exchanged between the service provider and a service requester ...);

- wherein the run time environment (e.g., [0108] – Figs. 5A-5D illustrate a flowchart of an exemplary method for processing ...) is arranged to implement the at least one node in the memory of the computer system, the node thereby becoming a memory resident node and the at least one memory resident node being arranged to process data according to the rules (e.g., Fig. 1, element – INVOKE; [0021] - ... invoke operations on web-services server ...; [0052] - ... specifies elements or data types that may be used for the application specific content ...; [0076] - ... A value of 'encoded' for the use attribute indicates that rules are used for encoding and the sender of the commitment receipt should obey the encoded rules; [0077]) wherein at least one of the rules is triggered if predetermined data is present and an output is generated from that memory resident node (e.g., [0080] - ... An attachment extension in accordance with schema 46 may be specified in predefined locations in the WSDL document, for example 'binding/operation/output' ...)

Further, Beringer discloses a web-services interface for a web-service comprises a message binding extension element operable to specify a format for a message requesting the web service and a binding details extension element operable to specify an availability status of at least one receipt for the message (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Exposing Process Flows and Choreography Controllers as Web Services*, Kasi discloses:

- the run time environment also comprises comprising a linker (e.g., Col. 4, Line 63 through Col. 5, Line 17 – Connectors may be labeled simple connectors, hubs or central connectors ... So-called hubs are used by connectors that are explicitly directed or linked to them ...; Col. 10, Lines 4-22 - ... a dispatcher in the connector can link the message to an existing process instance ...) which is arranged to connect, dynamically according to the specification and/or any data input to the application (e.g., Col. 7, Lines 37-43 – ... to dynamically determine some or all of document versions, service version, and choreography version ... a web service can be substantially updated and retain interoperability with other services that have not been similarly updated), the at least one node to any other nodes such that data input to the application is processed by the at least one node (e.g., Col. 6, Lines 31-32 – Message are sent and received by service instances; Col. 8, Line 61 through Col. 9, Line 17 – The input or output to an activity is called message. A message consists of multiple parts ... The part type could be an

XML part, or a part that is described by MIME type ...) and, if further processing is required, forwarded to the other nodes for that further processing (e.g., Col. 9, Lines 27-29 – This context goes through a series of state transitions triggered by other events and the process itself can initiate events as part of the flow; Col. 5, Lines 18-31 - ... defines the input/output messages of the operations ... synchronous, with one inward message followed by one outward message ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Kasi into the Beringer's system to further provide other limitations stated above in the Beringer system.

The motivation is that it would further enhance the Beringer's system by taking, advancing and/or incorporating the Kasi's system which offers significant advantages for supporting document exchange choreographies; and facilitating evolution of systems by various combinations of choreography versioning, service versioning and document versioning as once suggested by Kasi (e.g., Abstract)

17. **As to claim 64** (Previously Presented) (incorporating the rejection in claim 63), refer to claim **63** above accordingly.

18. **As to claim 65** (Previously Presented) (incorporating the rejection in claim 1), Kasi discloses the method wherein the or each node is arranged to manipulate data contained in a message (e.g., Col. 7, Lines 59-60 – The services correlate messages

using application logic that examines the message payloads; Col. 10, Lines 4-26 - ... by deducing the instance based on payload content ...)

19. **As to claim 67** (Previously Presented) (incorporating the rejection in claim 1), Kasi discloses the method wherein the or each node is arranged, during processing of the machine readable data file, to output data to any of the nodes to which it is arranged to be connected (e.g., Col. 4, Line 63 through Col. 5, Line 17 – Connectors may be labeled simple connectors, hubs or central connectors ... So-called hubs are used by connectors that are explicitly directed or linked to them ...; Col. 10, Lines 4-22 - ... a dispatcher in the connector can link the message to an existing process instance ...)

20. **As to claim 68** (New) (incorporating the rejection in claim 1), Beringer discloses the method wherein the run time environment directly processes the specification held in the machine readable data file (e.g., [0003] – WSDL (Web Services Description Language) is a web-services description language that describes web-services by specifying parts, message, operations, ports, port types and services. It comprises an XML (eXtensible Markup Language) vocabulary that standardizes how organizations describe web-services ...)

21. **As to claim 69** (new) (incorporating the rejection in claim 1), Beringer discloses the method wherein the run time environment processes the specification held in the machine readable data file without the need for extra processing of the specification (e.g., [0003] – WSDL (Web Services Description Language) is a web-services

description language that describes web-services by specifying parts, message, operations, ports, port types and services. It comprises an XML (eXtensible Markup Language) vocabulary that standardizes how organizations describe web-services ...)

22. **As to claim 70** (New) (incorporating the rejection in claim 36), please refer to claim **69** above, accordingly.

23. **As to claim 71** (new) (incorporating the rejection in claim 63), please refer to claim **69** above, accordingly.

24. Claims 3-9 and 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Beringer in view of Kasi and Saga Software, Inc., (WO 00/29924) (hereinafter 'Saga')

25. **As to claim 3** (Previously Presented) (incorporating the rejection in claim 1), Further, Kasi discloses facilitating evolution of system by various combinations of choreography versioning (e.g., Abstract) but Beringer and Kasi do not explicitly disclose the limitations stated below.

However, in an analogous art of *Extensible Distributed Enterprise Application Integration System*, Saga discloses the method which further comprises providing a library of nodes containing at least one node and selecting at least one of the nodes from the library of nodes (e.g., Fig. 3, element 132 – Class Libraries)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Saga into the Beringer-Kasi's system to further provide the limitations stated above in the Beringer-Kasi system.

The motivation is that it would further enhance the Beringer-Kasi's system by taking, advancing and/or incorporating the Saga's system which offers significant advantages that this agent-adapter architecture is its ability to host complex business logic in order to maintain state and negotiate transactions with the application resources as once suggested by Saga (e.g., P. 21, Lines 7-14)

26. **As to claim 4** (Previously Presented) (incorporating the rejection in claim 1), Saga discloses the method which further comprises arranging the or each node to comprise a plurality of layers, each layer being arranged to perform a predetermined function (e.g., P. 15, Lines 18-20 - an "Enterprise Messaging Service (EMS)" ....; P. 17, Lines 6-8 - "Message-Oriented Middleware (MON)" ... to enable application on the same or different platforms to communicate ...; P. 20, Lines 2-3 - ... including class libraries, wizards, and templates ...)

27. **As to claim 5** (Previously Presented) (incorporating the rejection in claim 4), Saga discloses the method which further comprises arranging the layers of the nodes to be interchangeable and wherein altering at least one of the layers can change the overall functionality of a node (e.g., P. 3, Lines 28-31 - ... message oriented middleware

(MOM) ... one-to-one application connectivity with MOM ... in many-to-many application integration ...)

28. **As to claim 6** (Previously Presented) (incorporating the rejection in claim 4), Saga discloses the method which further comprises providing a library of layers containing at least one layer and selecting at least one layer from the library of layers (e.g., Fig. 3, element 132 – Class Libraries)

29. **As to claim 7** (Previously Presented) (incorporating the rejection in claim 4), Saga discloses the method which comprises arranging at least one of the layers of a node to act as a transport layer arranged to receive and send data to and from the node (e.g., P. 17, Lines 6-8 – "Message-Oriented Middleware (MON)" ... to enable applications on the same or different platforms to communicate ...)

30. **As to claim 8** (Previously Presented) (incorporating the rejection in claim 4), Saga discloses the method which comprises arranging at least one of the layers of a node to act as a message transceiver arranged to send and receive messages to other nodes to which that node is connected (e.g., P. 15, Lines 18-20 – an "Enterprise Messaging Service (EMS)" ... using the Java® Messaging Server (JMS). It enables system to use multiple messaging modes, and supports message hubs and provides message persistence)

31. **As to claim 9** (Previously Presented) (incorporating the rejection in claim 8), Saga discloses the method in which the at least one node has an identity and in which the application is arranged to be run and, at runtime, as nodes are connected together, the method further comprising arranging the message transceiver layer of a node to discover the identity of nodes to which it is connected at runtime (e.g., Figs. 10(a) and 10(b); P. 23, Lines 32-34 - ... message hubs are used to receive system messages ... and to hold those system messages .. can deliver same to one or more target integration objects; P. 3, Lines 28-31 - ... message oriented middleware (MOM) ... one-to-one application connectivity with MOM ... in many-to-many application integration ...)

32. **As to claim 35** (Previously Presented) (incorporating the rejection in claim 1), Saga discloses the method in which at least one node is provided to provide an output from the application (e.g., P. 17, Lines 31-33 - ... a "primary input message" is the main input data to the system transformation processes specified in transformer definitions. The system takes input data, transforms it, and creates output data needed by target applications)

33. Claims 20-26, 28-34, 42-48, 62, and 66 are rejected under 35 U.S.C. 103(a) as being unpatentable over Beringer in view of Kasi and Thilmany et al. (*BizTalk®: Implement Design Patterns for Business Rules with Orchestration Designer, Oct. 2001, MSDN® Magazine*) (hereinafter 'Thilmany')

34. **As to claim 20** (Previously Presented) (incorporating the rejection in claim 1), Beringer and Kasi do not explicitly disclose the limitations stated below.

However, in an analogous art of *Implement Design Patterns for Business Rules with Orchestration Designer*, Thilmany discloses the method which comprises providing a pattern, arranging the at least one node within the pattern and defining how nodes therein interact with one another (e.g., SUMMARY – this article describes several traditional design patterns including the Observer pattern and the Dispatcher pattern, elaborates on their structures, what they're used for, and how they can help you build a Biztalk®-based solution; Sec. of The Chain of Responsibility Pattern, 1st Para – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Thilmany into the Beringer-Kasi's system to further provide the limitations stated above in the Beringer-Kasi's system.

The motivation is that it would further enhance the Beringer-Kasi's system by taking, advancing and/or incorporating Thilmany's system which offers significant advantages that the sample application will show how applying patterns to BizTalk® can significantly ease the development and design for each sub-department as once suggested by Thilmany (e.g., P. 1, 5th Para)

35. **As to claim 21** (Previously Presented) (incorporating the rejection in claim 20), Thilmany discloses the method which comprises providing a library of patterns containing at least one pattern that can be used in creating an application (e.g., SUMMARY – this article describes several traditional design patterns including the Observer pattern and the Dispatcher pattern, elaborates on their structures, what they're used for, and how they can help you build a Biztalk-based solution; Sec. of The Chain of Responsibility Pattern, 1st Para – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Design Patterns versus Architectural Patterns, 1st Para – Reusability not only applies to the components themselves, but also to the stages the design must go through to morph itself into your final solution; the ability to apply a patterned repeatable solution is worth the little time spent learning formal patterns, or to even formalize you own)

36. **As to claim 22** (Previously Presented) (incorporating the rejection in claim 1), Thilmany discloses the method in which the specification is arranged to determine at least one of the following: which nodes are to be used; which nodes interact with one another; which patterns are to be used; which assets are to be used (e.g., SUMMARY – this article describes several traditional design patterns including the Observer pattern and the Dispatcher pattern, elaborates on their structures, what they're used for, and

how they can help you build a Biztalk-based solution; Sec. of The Chain of Responsibility Pattern, 1st Para – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of The Chain of Responsibility Pattern, 2nd Para – as is the case of our product support sample, the request may not be fulfilled at one application site due to the distribution of the knowledge bases used to answer a request; the request must be routed until it can be handled)

37. **As to claim 23** (Previously Presented) (incorporating the rejection in claim 1), Thilmany discloses the method which comprises providing files arranged to define the application specified therein, arranging the specification to be capable of deploying files and using the specification to deploy the files (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Para – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML)

38. **As to claim 24** (Previously Presented) (incorporating the rejection in claim 23), Thilmany discloses the method which comprises arranging the files specifying the application to be XML files (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Para – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design

tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML)

39. **As to claim 25** (Previously Presented) (incorporating the rejection in claim 1), Thilmany discloses the method in which the data processed by the application is specified in an XML file (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Para – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML)

40. **As to claim 26** (Previously Presented) (incorporating the rejection in claim 1), Thilmany discloses the method in which data processed by the application is specified in an image file (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer, analysts not privy to the implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio® to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate)

41. **As to claim 28** (Previously Presented) (incorporating the rejection in claim 1), Thilmany discloses the method which comprises providing a graphical tool arranged to enable a user to specify components of the application (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer, analysts not privy to the implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio® to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; Sec. of BizTalk® and the Orchestration Designer®, 3rd Para – the Biztalk® Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML)

42. **As to claim 29** (Previously Presented) (incorporating the rejection in claim 28), Thilmany discloses the method which comprises providing a library of at least one of the following: nodes; node layers; specification; patterns; messages; rule sets; style sheets; schemas and in which the graphical tool allows a user to select components from one of said libraries (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration

Designer, analysts not privy to the implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio® to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; SUMMARY – this article describes several traditional design patterns including the Observer pattern and the Dispatcher pattern, elaborates on their structures, what they're used for, and how they can help you build a Biztalk®-based solution; Sec. of The Chain of Responsibility Pattern, 1st Para – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of The Chain of Responsibility Pattern, 2nd Para – as is the case of our product support sample, the request may not be fulfilled at one application site due to the distribution of the knowledge bases used to answer a request; the request must be routed until it can be handled)

43. **As to claim 30** (Previously Presented) (incorporating the rejection in claim 29), Thilmany discloses the method which allows a user to define further libraries (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer®, analysts not privy to the implementation complexities of COM or Web development can participate in the

development of business rules by using the designer in Visio to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; Sec. of BizTalk® and the Orchestration Designer®, 3rd Para – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML)

44. **As to claim 31** (Previously Presented) (incorporating the rejection in claim 28), Thilmany discloses the method which comprises providing at least one pattern arranged to define how nodes interact arranging the at least one pattern such that it is capable of interacting with at least one other pattern and arranging the graphical tool to allow a user to specify how the patterns and nodes interact with one another (e.g., SUMMARY – this article describes several traditional design patterns including the Observer pattern and the Dispatcher pattern, elaborates on their structures, what they're used for, and how they can help you build a Biztalk®-based solution; Sec. of The Chain of Responsibility Pattern, 1st Para – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of The Chain of Responsibility Pattern, 2nd Para – as is the case of our product support sample, the request may not be fulfilled at one application site due to the distribution of the knowledge bases used to answer a request; the request must be routed until it can be handled)

45. **As to claim 32** (Previously Presented) (incorporating the rejection in claim 28), Thilmany discloses the method which comprises using the graphical tool to perform at least one of the following: create the specification; edit the specification (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer, analysts not privy to the implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; Sec. of BizTalk® and the Orchestration Designer®, 3rd Para – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML)

46. **As to claim 33** (Previously Presented) (incorporating the rejection in claim 28), Thilmany discloses the method which comprises using the graphical tool to manipulate any components of the specification (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer®, analysts not privy to the implementation complexities

of COM or Web development can participate in the development of business rules by using the designer in Visio® to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; Sec. of BizTalk® and the Orchestration Designer®, 3rd Para – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML)

47. **As to claim 34** (Previously Presented) (incorporating the rejection in claim 1), Thilmany discloses the method which comprises creating and deploying files and processing the files in the run time environment (e.g., Sec. of Biztalk® and the Orchestration Designer®, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of message quest, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; with the Orchestration Designer®, analysts not privy to the implementation complexities of COM or Web development can participate in the development of business rules by using the designer in Visio to lay out the business flow; the developer can then implement the moving parts that these designers orchestrate; Sec. of BizTalk® and the Orchestration Designer®, 3rd Para – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML)

48. **As to claim 42** (Previously Presented) (incorporating the rejection in claim 36), Thilmany discloses the computer system in which the deployer deploys the definition that causes the nodes to communicate with one another using one of HTTP and direct memory protocols (e.g., Sec. of the chain of responsibility pattern, 1st Para – the intent of the Chain of Responsibility pattern is to provide a loosely coupled sender/receiver relationship by providing more than one component an opportunity to handle a request; this chains all receiving components and passes the request along the chain until the request is handled; Sec. of Using the BizTalk Orchestration Designer®, 2nd Para – ports are named locations where messages are sent and received; ports can be defined initially as unbound or bound; Sec. of BizTalk and the Orchestration Designer®, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk Channel, e-mail message, or HTTP-based service; orchestration designer provides a way to create loosely coupled business processes that may optionally be long-running a nature)

49. **As to claim 43** (Previously Presented) (incorporating the rejection in claim 36), Thilmany discloses the computer system in which the node creation means creator is arranged to utilise at least one of the following: predetermined definitions and pre-written definitions (e.g., Sec. of Using the BizTalk Orchestration Designer®, 1st Para – there are three key object categories in the business process page of the Orchestration Designer®; flowchart shapes, ports, and implementation shapes; conceptually, the way

this works is that you, or your friendly neighborhood business analyst, use the flowchart shapes to create the process flow; the flowchart shapes support basic constructs such as if … then … else decisions, looping while a condition is true, branching execution, rejoining the branches together, and defining transactional boundaries)

50. **As to claim 44** (Previously Presented) (incorporating the rejection in claim 43), Thilmany discloses the computer system in which the pre-written definition is provided in at least one library (e.g., P. 1, 6th Para – the sample application will show how applying patterns to Biztalk can significantly ease the development and design for each sub-department; it will provide the ability to answer and route online product support questions effectively and efficiently; Sec. of Design Patterns versus Architectural Patterns, 1st Para – design patterns help make the design process faster; this allows solution providers to take the time to concentrate on the business implementation; more importantly, patterns help formalize the design to make it reusable; reusability not only applies to the components themselves, but also the stages the design must go through to morph itself into your final solution; Sec. of Biztalk and the Orchestration Designer®, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; Orchestration Designer provides a way to create loosely coupled business processes that may optionally be long-running in nature)

51. **As to claim 45** (Previously Presented) (incorporating the rejection in claim 36), Thilmany discloses the computer system which further comprises a pattern creation means creator arranged to create at least one pattern of nodes (e.g., P. 1, 6th Para – the sample application will show how applying patterns to Biztalk® can significantly ease the development and design for each sub-department; it will provide the ability to answer and route online product support questions effectively and efficiently; Sec. of Design Patterns versus Architectural Patterns, 1st Para – design patterns help make the design process faster; this allows solution providers to take the time to concentrate on the business implementation; more importantly, patterns help formalize the design to make it reusable; reusability not only applies to the components themselves, but also the stages the design must go through to morph itself into your final solution; Sec. of Biztalk® and the Orchestration Designer, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; Orchestration Designer provides a way to create loosely coupled business processes that may optionally be long-running in nature)

52. **As to claim 46** (Previously Presented) (incorporating the rejection in claim 36), Thilmany discloses the computer system which further comprises a pattern cloner arranged to clone a pattern of nodes (e.g., P. 1, 6th Para – the sample application will show how applying patterns to Biztalk can significantly ease the development and design for each sub-department; it will provide the ability to answer and route online

product support questions effectively and efficiently; Sec. of Design Patterns versus Architectural Patterns, 1st Para – design patterns help make the design process faster; this allows solution providers to take the time to concentrate on the business implementation; more importantly, patterns help formalize the design to make it reusable; reusability not only applies to the components themselves, but also the stages the design must go through to morph itself into your final solution; Sec. of Biztalk® and the Orchestration Designer®, 3rd Para – the designer provides a high-level means of orchestrating a middleware system's moving parts; those moving parts can take the form of a message queue, COM component, BizTalk® Channel, file, e-mail message, or HTTP-based service; Orchestration Designer provides a way to create loosely coupled business processes that may optionally be long-running in nature)

53. **As to claim 47** (Previously Presented) (incorporating the rejection in claim 36), Thilmany discloses the computer system which further comprises a rule creation means creator arranged to allow predetermined rules to be created and edited (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd Para – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML)

54. **As to claim 48** (Previously Presented) (incorporating the rejection in claim 36), Thilmany discloses the computer system which further comprises at least one of the following: a node storage (e.g., Sec. of BizTalk® and the Orchestration Designer®, 3rd

Para – the Biztalk Orchestration Designer® is a Microsoft Visio® 2000-based design tool that, when compiled, can run as an XLANG® schedule; XLANG® itself is described through XML); a pattern storage (e.g., P. 1, 6th Para – the sample application will show how applying patterns to Biztalk can significantly ease the development and design for each sub-department; it will provide the ability to answer and route online product support questions effectively and efficiently); a rule storage (e.g., Sec. of Using the BizTalk Orchestration Designer®, 1st Para – there are three key object categories in the business process page of the Orchestration Designer®; flowchart shapes, ports, and implementation shapes; conceptually, the way this works is that you, or your friendly neighborhood business analyst, use the flowchart shapes to create the process flow; the flowchart shapes support basic constructs such as if … then … else decisions, looping while a condition is true, branching execution, rejoining the branches together, and defining transactional boundaries)

55. **As to claim 62** (Previously Presented) (incorporating the rejection in claim 1), Thilmany discloses the method in which the node, specification, and messages are at least in part written in XML (e.g., Sec. of BizTalk® and the Orchestration Designer®, 2nd Para – communicating to a BizTalk® server implementation simply means using XML as the message format; as you may already know, BizTalk® is completely structured around XML and uses a SOAP 1.1 XML message format; the BizTalk® friendly XML message is nothing more than a SOAP 1.1 XML message with additional biz-tags to comply with the BizTalk® Framework specification)

56. **As to claim 66** (Previously Presented) (incorporating the rejection in claim 65), Thilmany discloses the method wherein the or each node is arranged to manipulate XML data contained in the message (e.g., Sec. of BizTalk® and the Orchestration Designer®, 2nd Para – communicating to a BizTalk® server implementation simply means using XML as the message format; as you may already know, BizTalk® is completely structured around XML and uses a SOAP 1.1 XML message format; the BizTalk® friendly XML message is nothing more than a SOAP 1.1 XML message with additional biz-tags to comply with the BizTalk® Framework specification)

57. Claims 10-12, and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Beringer in view of Kasi and Moore et al., (Pub. No. US 2004/0034848 A1 B1) (hereinafter 'Moore')

58. **As to claim 10** (Previously Presented) (incorporating the rejection in claim 4), Beringer and Kasi do not explicitly disclose the limitations stated below.

However, in an analogous art of *Rule Engine*, Moore discloses the method which comprises arranging at least one of the layers of a node to act as a rule processing engine arranged to apply the predetermined rules to data that the node receives (e.g., Fig. 1 – element of 140 – Business Intelligence Server; [0031], Lines 16-18 – rule-packs are implemented as extensible markup language (XML) documents expressed in a rules markup language generated for that purpose)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Moore into the Beringer-Kasi's system to further provide the limitations stated above in the Beringer-Kasi system.

The motivation is that it would further enhance the Beringer-Kasi's system by taking, advancing and/or incorporating Moore's system which offers significant advantages for automating business processes including, in a computer system, receiving a rule set as a single package, determining logical conflicts within the rule set, resolving the logical conflicts, and generating a sequence of processing logic from the rule set for optimal processing of inputted facts as once suggested by Moore (e.g., [0005])

59. **As to claim 11** (Previously Presented) (incorporating the rejection in claim 10), Moore discloses the method in which the rule processing engine layer of a node is arranged to use uses forward chaining rule logic (e.g., [0002], Lines 8-12 – forward chaining is a process of applying a set of previously determined rules to the facts in a knowledge base to see if any of them fire and thereby generate new facts; in essence, all derivable knowledge is derived in forward chaining)

60. **As to claim 12** (Previously Presented) (incorporating the rejection in claim 10), Moore discloses a method which comprises providing a rule set of at least one rule in a file that is used by the rule processing engine (e.g., [0005], automating business processes including, in a computer system, receiving a rule set as a single package,

determining logical conflicts within the rule set, resolving the logical conflicts, and generating a sequence of processing logic from the rule set for optimal processing of inputted facts)

61. **As to claim 14** (Previously Presented) (incorporating the rejection in claim 10), Moore discloses the method which comprises defining each rule set that is to be used by the application in the specification (e.g., [0031], Lines 16-18 – rules-packs are implemented as extensible markup language (XML) documents expressed in a rules markup language generated for that purpose)

62. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Beringer in view of Kasi, Moore and further in view of Lee et al. (*The Extensible Rule Markup Language, May 2003, ACM*) (hereinafter 'Lee')

63. **As to claim 13** (Previously Presented) (incorporating the rejection in claim 12), Beringer, Kasi and Moore do not explicitly disclose the limitations stated below.

However, in an analogous art of *The Extensible Rule Markup Language*, Lee discloses the method which comprises specifying the file in which the rules are located by a link (e.g., P. 60, Left-Col, "Relevance Linkability" – Linkages of the relevance between hypertexts with RIML, as well as rules in RSML syntax (called RSML rules), should be expressed completely) .

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Lee into the Beringer-Kasi-Moore's system to further provide the limitations stated above in the Beringer-Kasi-Moore's system.

The motivation is that it would further enhance the Beringer-Kasi-Moore's system by taking, advancing and/or incorporating Lee's system which offers significant advantages that the Extensible Markup Language (XML) explicates the implicitly embedded data in a formal structure with mutually agreed-on semantic definitions; may industrial-strength standard initiatives using XML are under way, including the Electronic Business XML Initiative, XML Common Business Library, Open Trading Protocol, Open Business on the Internet, Common Business Language, RosettaNet, and Biztalk as once suggested by Lee (e.g., P. 59, 1st Para)

Conclusion

64. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Ben C. Wang
Examiner, Art Unit 2192

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192